

10C9  
10-  
10-  
10-  
10-  
10-  
10-  
KICL

KILL  
KILL  
LB\_E  
LB\_C  
LB\_F  
LB\_H  
LB\_L  
LOCA  
LOCA  
LOCK

LOCK  
LOCK  
LOCK  
LOC-  
LOC-  
L-CC  
L-CC  
L-DA  
L-DA  
MAIN  
MAKE  
MAKE  
MAKE  
MAKE  
MAKE  
MAKE

MAKE  
MAKE  
MAP  
MAP

MAP  
MAR  
MAR  
MAR  
MAR  
MAR

FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFFFFFFFF.FFF	111	111		
FFFFFFFFFFFFFFFF	111	111	XXX	
FFFFFFFFFFFFFFFF	111	111	XXX	
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	1111111111	1111111111	XXX	XXX
FFF	1111111111	1111111111	XXX	XXX
FFF	1111111111	1111111111	XXX	XXX

```
TTTTTTTTT1  RRRRRRRR  UU      UU  NN      NN  CCCCCCCC
TTTTTTTTT1  RRRRRRRR  UU      UU  NN      NN  CCCCCCCC
TT          RR      RR  UU      UU  NN      NN  CC
TT          RR      RR  UU      UU  NN      NN  CC
TT          RR      RR  UU      UU  NNNN     NN  CC
TT          RR      RR  UU      UU  NNNN     NN  CC
TT          RRRRRRRR  UU      UU  NN      NN  CC
TT          RRRRRRRR  UU      UU  NN      NN  CC
TT          RR  RR    UU      UU  NN      NN  CC
TT          RR  RR    UU      UU  NN      NN  CC
TT          RR      RR  UU      UU  NN      NN  CC
TT          RR      RR  UUUUUUUUUU  NN      NN  CCCCCCCC
TT          RR      RR  UUUUUUUUUU  NN      NN  CCCCCCCC
```

```
LL          IIIIII  SSSSSSSS
LL          IIIIII  SSSSSSSS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SSSSSS
LL          II      SSSSSS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SS
LLLLLLLLLLL IIIIII  SSSSSSSS
LLLLLLLLLLL IIIIII  SSSSSSSS
```



```
1 0001 0 MODULE TRUNC (
2 0002 0
3 0003 0 LANGUAGE (BLISS32),
4 0004 0 IDENT = 'V04-000'
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: F11ACP Structure Level 2
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 This routine truncates a file by deallocating the indicated blocks.
38 0038 1
39 0039 1 ENVIRONMENT:
40 0040 1
41 0041 1 STARLET operating system, including privileged system services
42 0042 1 and internal exec routines.
43 0043 1
44 0044 1 --
45 0045 1
46 0046 1
47 0047 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 21-Mar-1977 10:41
48 0048 1
49 0049 1 MODIFIED BY:
50 0050 1
51 0051 1 V03-013 CDS0007 Christian D. Saether 14-Aug-1984
52 0052 1 Remove reference to update_filesize routine.
53 0053 1
54 0054 1 V03-012 CDS0006 Christian D. Saether 31-July-1984
55 0055 1 Remove local declaration of get_map_pointer linkage.
56 0056 1
57 0057 1 V03-011 CDS0005 Christian D. Saether 5-July-1984
```

```

58 0058 1 Do not call READ_HEADER with the file id argument
59 0059 1 when re-reading primary header at the end because
60 0060 1 we always have a primary fcb now and when this
61 0061 1 routine is called from deaccess on a deferred
62 0062 1 truncate, the fid field is not filled in.
63 0063 1
64 0064 1 V03-010 CDS0004 Christian D. Saether 22-Apr-1984
65 0065 1 Change linkage L_TRUNC_CHECKS to L_JSB_2ARGS.
66 0066 1
67 0067 1 V03-009 CDS0003 Christian D. Saether 30-Dec-1983
68 0068 1 Use L_NORM linkage and BIND_COMMON macro.
69 0069 1
70 0070 1 V03-008 CDS0002 Christian D. Saether 25-Sep-1983
71 0071 1 Manually merge in code associated with STJ3097.
72 0072 1
73 0073 1 V03-007 STJ3097 Steven T. Jeffreys, 29-Apr-1983
74 0074 1 Refinement of STJ3072. Only do the erase if the
75 0075 1 volume or file ERASE attribute is set.
76 0076 1
77 0077 1 V03-006 CDS0001 Christian D. Saether 21-Apr-1983
78 0078 1 Break out initial error checks into separate routine.
79 0079 1 Make the truncation vbn an input argument.
80 0080 1
81 0081 1 V03-005 STJ3072 Steven T. Jeffreys, 25-Mar-1983
82 0082 1 Erase blocks returned to the storage map. Later this
83 0083 1 will be conditionalized.
84 0084 1
85 0085 1 V03-004 ACG0299 Andrew C. Goldstein, 12-Oct-1982 17:21
86 0086 1 Make truncate tolerant of bad map pointer use count
87 0087 1
88 0088 1 V03-003 ACG0296 Andrew C. Goldstein, 8-Jul-1982 21:32
89 0089 1 Fix truncation of placed allocation pointers
90 0090 1
91 0091 1 V03-002 ACG0287 Andrew C. Goldstein, 14-Apr-1982 17:16
92 0092 1 Check for index file in header rather than FCB
93 0093 1
94 0094 1 V03-001 LMP0023 L. Mark Pilant, 7-Apr-1982 16:45
95 0095 1 Give a privilege violation if attempting to truncate the
96 0096 1 index file (INDEXF.SYS).
97 0097 1
98 0098 1 V02-011 ACG35898 Andrew C. Goldstein, 10-Mar-1981 22:15
99 0099 1 Update HIBLK in the primary header
100 0100 1
101 0101 1 V02-010 ACG0170 Andrew C. Goldstein, 7-May-1980 18:27
102 0102 1 Fix handling of map pointer use count
103 0103 1
104 0104 1 V02-009 ACG0167 Andrew C. Goldstein, 16-Apr-1980 19:28
105 0105 1 Previous revision history moved to F11B.REV
106 0106 1 **
107 0107 1
108 0108 1
109 0109 1 LIBRARY 'SYSS$LIBRARY:LIB.L32';
110 0110 1 REQUIRE 'SRC$:FCPDEF.B32';
111 1101 1
112 1102 1
113 1103 1 FORWARD ROUTINE
114 1104 1 TRUNCATE : L_NORM NOVALUE, ! truncate file
```



TRUNC  
V04-000

K 10  
16-Sep-1984 01:19:12  
14-Sep-1984 12:30:50

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[F11X.SRC]TRUNC.B32;1 Page 3 (1)

: 115  
: 116

1105 1  
1106 1

TRUNCATE HEADER : L\_NORM NOVALUE, ! truncate individual file header  
TRUNC\_CHECKS : L\_JSB\_2ARGS NOVALUE ; ! initial truncation error checks.

```
118 1107 1 GLOBAL ROUTINE TRUNCATE (FIB, FILEHEADER, TRNVBN) : L_NORM NOVALUE =
119 1108 1
120 1109 1 !++
121 1110 1
122 1111 1 FUNCTIONAL DESCRIPTION:
123 1112 1
124 1113 1 This routine truncates a file to the size indicated in the FIB by
125 1114 1 deallocating the necessary blocks. The erase flag controls whether
126 1115 1 the retrieval pointers are erased in the header. The deallocate flag
127 1116 1 controls whether or not the blocks are actually returned to the
128 1117 1 storage map.
129 1118 1
130 1119 1 CALLING SEQUENCE:
131 1120 1 TRUNCATE (ARG1, ARG2, ARG3)
132 1121 1
133 1122 1 INPUT PARAMETERS:
134 1123 1 ARG1: address of FIB for operation
135 1124 1 ARG2: address of file header
136 1125 1 ARG3: VBN to truncate from
137 1126 1
138 1127 1 IMPLICIT INPUTS:
139 1128 1 CURRENT_VCB: VCB of volume
140 1129 1
141 1130 1 OUTPUT PARAMETERS:
142 1131 1 NONE
143 1132 1
144 1133 1 IMPLICIT OUTPUTS:
145 1134 1 NONE
146 1135 1
147 1136 1 ROUTINE VALUE:
148 1137 1 NONE
149 1138 1
150 1139 1 SIDE EFFECTS:
151 1140 1 storage bitmap altered
152 1141 1 file header altered
153 1142 1
154 1143 1 --
155 1144 1
156 1145 2 BEGIN
157 1146 2
158 1147 2 MAP
159 1148 2 FIB : REF BBLOCK, ! FIB for operation
160 1149 2 FILEHEADER : REF BBLOCK; ! file header
161 1150 2
162 1151 2 LINKAGE
163 1152 2 L_MAKE_POINTER = CALL :
164 1153 2 GLOBAL (PREV_POINTER = 9);
165 1154 2
166 1155 2 GLOBAL REGISTER
167 1156 2 COUNT = 6, ! count of blocks returned
168 1157 2 LBN = 7, ! LBN of map entry
169 1158 2 MAP_POINTER = 8 : REF BBLOCK, ! pointer to scan map
170 1159 2 PREV_POINTER = 9 : REF BBLOCK; ! pointer to build new map entry
171 1160 2
172 1161 2 LOCAL
173 1162 2 FCB : REF BBLOCK, ! FCB of current file header
174 1163 2 HEADER : REF BBLOCK, ! address of current file header
```



```
175      1164 2      ALT_HEADER      : REF BBLOCK,      ! address of header copy to free blocks
176      1165 2      NEW_HEADER      : REF BBLOCK,      ! address of extension file header
177      1166 2      TRUNC_POINTER,    ! pointer to start of truncation
178      1167 2      MAP_END,          ! pointer to end of map area
179      1168 2      VBN,              ! relative VBN of operation
180      1169 2      HEADER_VBN,       ! value of VBN at start of this header
181      1170 2      HEADER_SIZE,      ! number of blocks mapped by header
182      1171 2      EXT_FID           : BBLOCK [FID$C_LENGTH], ! file ID of extension header
183      1172 2      EX_SEGNUM,        ! segment number of ext header
184      1173 2      REREAD,           ! flag to reread primary header
185      1174 2      REREAD2;          ! flag to update primary header
186      1175 2
187      1176 2 LABEL
188      1177 2      DO TRUNCATE,       ! main body of truncate processing code
189      1178 2      VBN_LOOP;          ! main loop to scan for starting VBN
190      1179 2
191      1180 2 BIND_COMMON;
192      1181 2
193      1182 2 EXTERNAL ROUTINE
194      1183 2      PMS_START SUB      : L_NORM,          ! start subfunction metering
195      1184 2      PMS_END SUB        : L_NORM,          ! end subfunction metering
196      1185 2      FILE_SIZE         : L_NORM,          ! compute size mapped by header
197      1186 2      SEARCH_FCB        : L_NORM,          ! search FCB list for FCB
198      1187 2      CHARGE_QUOTA       : L_NORM,          ! charge blocks to user's quota
199      1188 2      DEALLOCATE_BAD     : L_NORM ADDRESSING_MODE (GENERAL),
200      1189 2      MARK DIRTY         : L_NORM,          ! mark blocks bad
201      1190 2      CHECKSUM           : L_NORM,          ! mark buffer for write-back
202      1191 2      GET_MAP_POINTER    : L_NORM,          ! checksum file header
203      1192 2      MAKE_POINTER       : L_NORM,          ! get value of next map pointer
204      1193 2      NEXT_HEADER        : L_NORM,          ! build a new map pointer
205      1194 2      CREATE_BLOCK       : L_NORM,          ! read next extension header
206      1195 2      INVALIDATE        : L_NORM,          ! allocate a block buffer
207      1196 2      INIT_FCB2         : L_NORM,          ! invalidate a block buffer
208      1197 2      WRITE_HEADER       : L_NORM,          ! initialize FCB
209      1198 2      READ_HEADER        : L_NORM,          ! write file header
210      1199 2      DEL_EXTFCB        : L_NORM,          ! read file header
211      1200 2      DELETE_FILE        : L_NORM,          ! delete extension FCB's
212      1201 2      DELETE_FILE        : L_NORM,          ! delete remainder of file
213      1202 2
214      1203 2
215      1204 2 ! Start metering for this subfunction.
216      1205 2
217      1206 2
218      1207 2 PMS_START_SUB (PMS_ALLOC);
219      1208 2
220      1209 2 TRUNC_CHECKS (.FIB, .FILEHEADER);
221      1210 2
222      1211 2 ! Establish the basic pointers. Round up the starting VBN to the next cluster
223      1212 2 ! boundary and adjust it to a zero start.
224      1213 2 ! Round down the file size.
225      1214 2
226      1215 2
227      1216 2 HEADER = .FILEHEADER;
228      1217 2 FCB = .PRIMARY_FCB;
229      1218 2 VBN = .TRNVBN = 1;
230      1219 2
231      1220 2 ! Init the user's return parameters.
```

```
232 1221 2 !
233 1222 3
234 1223 2 FIB[FIB$L_EXVBN] = 1;
235 1224 2
236 1225 2 REREAD = 0;
237 1226 2
238 1227 2 ! Now scan the file headers for the retrieval pointer containing the starting
239 1228 2 ! VBN. If the VBN is off the end of file, report the error; if it coincides,
240 1229 2 ! the operation is a noop.
241 1230 2 !
242 1231 2
243 1232 2 DO TRUNCATE:
244 1233 3 BEGIN
245 1234 3
246 1235 3 VBN_LOOP:
247 1236 4 BEGIN
248 1237 4 WHILE 1 DO
249 1238 5 BEGIN
250 1239 5 MAP_POINTER = .HEADER + .HEADER[FH2$B_MPOFFSET]*2;
251 1240 5 MAP_END = .MAP_POINTER + .HEADER[FH2$B_MAP_INUSE]*2;
252 1241 5 PREV_POINTER = .MAP_POINTER;
253 1242 5 HEADER_VBN = .VBN;
254 1243 5
255 1244 5 UNTIL .MAP_POINTER GEQA .MAP_END DO
256 1245 6 BEGIN
257 1246 6 GET_MAP_POINTER ();
258 1247 6 IF .COUNT GEQU .VBN THEN LEAVE VBN_LOOP;
259 1248 6 VBN = .VBN - .COUNT;
260 1249 6 FIB[FIB$L_EXVBN] = .FIB[FIB$L_EXVBN] + .COUNT;
261 1250 6 IF .COUNT NEQ 0
262 1251 6 THEN PREV_POINTER = .MAP_POINTER;
263 1252 5 END;
264 1253 5
265 1254 5 ! We have scanned through an entire header. Chain to the next header if it
266 1255 5 ! exists. If we run out of headers, then the truncate point is beyond end
267 1256 5 ! of file.
268 1257 5 !
269 1258 5
270 1259 5 NEW_HEADER = NEXT_HEADER (.HEADER, .FCB);
271 1260 5 IF .NEW_HEADER EQ 0 THEN EXITLOOP;
272 1261 5 REREAD = 1;
273 1262 5 HEADER = .NEW_HEADER;
274 1263 5
275 1264 5 IF .FCB NEQ 0
276 1265 5 THEN FCB = .FCB[FCB$L_EXFCB];
277 1266 4 END; ! end of header scan loop
278 1267 4
279 1268 4 IF .VBN NEQ 0
280 1269 5 THEN ERR_EXIT (SS$_ENDOFFILE)
281 1270 3 END; ! end of VBN_LOOP
282 1271 3
283 1272 3 ! We are now pointing at the retrieval pointer in which the truncation starts.
284 1273 3 ! VBN contains the number of blocks to retain in that pointer. We must now
285 1274 3 ! round it down or up to the next cluster boundary, depending on whether or
286 1275 3 ! not the blocks are to be marked bad, respectively.
287 1276 3 !
288 1277 3
```



```
1278 3 USER_STATUS[1] = - .VBN;  
1279 5 VBN = ((.VBN  
1280 6 + (IF NOT .FIB[FIB$V_MARKBAD]  
1281 6 THEN .CURRENT_VCB[VCB$W_CLUSTER] - 1  
1282 6 ELSE 0)  
1283 6 )  
1284 6 / .CURRENT_VCB[VCB$W_CLUSTER]) * .CURRENT_VCB[VCB$W_CLUSTER];  
1285 3 USER_STATUS[1] = .USER_STATUS[1] + .VBN;  
1286 3 FIB[FIB$L_EXVBN] = .FIB[FIB$L_EXVBN] + .VBN;  
1287 3 HEADER_VBN = .HEADER_VBN + .USER_STATUS[1];  
1288 3  
1289 3 ! See if rounding up is causing us to keep the entire map pointer. If so,  
1290 3 ! bump to the next pointer. If that takes us to the end of the map area of  
1291 3 ! a header with no extension, return with no action. (This case is common  
1292 3 ! enough to be worth checking for.)  
1293 3 !  
1294 3  
1295 3 IF .VBN EQL .COUNT  
1296 3 THEN  
1297 4 BEGIN  
1298 4 VBN = 0;  
1299 4 PREV_POINTER = .MAP_POINTER;  
1300 4 IF .PREV_POINTER GEQA .MAP_END  
1301 4 AND .HEADER[FH2$W_EX_FIDNUM] EQL 0  
1302 4 AND .HEADER[FH2$W_EX_FIDRVN] EQL 0  
1303 4 THEN LEAVE DO_TRUNCATE;  
1304 4 END;  
1305 3  
1306 3 ! If we are turning blocks over to the bad block file, check that  
1307 3 ! (1) the pointer given is the last pointer in the header,  
1308 3 ! (2) the header is the last one for the file, and (3) that the quantity  
1309 3 ! being deallocated is exactly one cluster, this being the only condition  
1310 3 ! we can correctly handle.  
1311 3 !  
1312 3  
1313 3 IF .FIB[FIB$V_MARKBAD]  
1314 3 THEN  
1315 4 IF .MAP_POINTER NEQ .MAP_END  
1316 4 OR .COUNT - .VBN NEQ .CURRENT_VCB[VCB$W_CLUSTER]  
1317 4 OR .HEADER[FH2$W_EX_FIDNUM] NEQ 0  
1318 4 OR .HEADER[FH2$W_EX_FIDSEQ] NEQ 0  
1319 4 THEN ERR_EXIT (SS$_BADPARAM);  
1320 3  
1321 3 ! Do the real truncate. Set up cleanup status and get control blocks in shape.  
1322 3 !  
1323 3  
1324 3 CLEANUP_FLAGS[CLF_FIXFCB] = 1;  
1325 3 CLEANUP_FLAGS[CLF_INVWINDOW] = 1;  
1326 3 PRIMARY_FCB [FCB$_FILESIZE] = .FIB[FIB$L_EXVBN] - 1;  
1327 3  
1328 3 ! Update the HIBLK field in the record attributes to reflect the new file  
1329 3 ! size, if this is the primary header..  
1330 3 !  
1331 3  
1332 3 IF NOT .REREAD  
1333 3 THEN BBLOCK [HEADER[FH2$W_RECATTR], FAT$L_HIBLK] = ROT (.FIB[FIB$L_EXVBN]-1, 16);  
1334 3
```



```
346 1335 3 ! Make a copy of the file header with which to free the blocks. In the original,
347 1336 3 ! zero out the map pointers being freed and write the header back before
348 1337 3 ! deallocating the blocks, so that we do not get a file header mapping free
349 1338 3 ! blocks if the system crashes while this is going on.
350 1339 3
351 1340 3
352 1341 3 ALT_HEADER = CREATE_BLOCK (-1, 1, HEADER_TYPE);
353 1342 3 INVALIDATE (.ALT_HEADER);
354 1343 3 CHSMOVE (512, .HEADER, .ALT_HEADER);
355 1344 3 TRUNC_POINTER = .PREV_POINTER - .HEADER + .ALT_HEADER;
356 1345 3
357 1346 3 HEADER[FH2$B_MAP_INUSE] = (.PREV_POINTER - .HEADER) / 2 - .HEADER[FH2$B_MPOFFSET];
358 1347 3 IF .VBN NEQ 0
359 1348 3 THEN
360 1349 3 BEGIN
361 1350 3 MAP_POINTER = .PREV_POINTER;
362 1351 3 GET_MAP_POINTER ();
363 1352 3 MAKE_POINTER (.VBN, .LBN, .HEADER,
364 1353 3 (IF .PREV_POINTER[FH2$V_FORMAT] EQL FM2$C_PLACEMENT
365 1354 3 THEN .PREV_POINTER[FH2$W_WORD0]
366 1355 3 ELSE 0));
367 1356 3 END;
368 1357 3 MAP_END = .HEADER + .HEADER[FH2$B_ACOFFSET]*2;
369 1358 3 IF .MAP_END - .PREV_POINTER GTR 0
370 1359 3 THEN CH$FILL (0, .MAP_END - .PREV_POINTER, .PREV_POINTER);
371 1360 3
372 1361 3 EX_SEGNUM = .HEADER[FH2$W_SEG_NUM] + 1;
373 1362 3 CHSMOVE (FID$C_LENGTH, HEADER[FH2$W_EXT_FID], EXT_FID);
374 1363 3 CH$FILL (0, FID$C_LENGTH, HEADER[FH2$W_EXT_FID]);
375 1364 3 CHECKSUM (.HEADER);
376 1365 3 WRITE_HEADER ();
377 1366 3
378 1367 3 IF .FCB NEQ 0 AND .FCB NEQ .PRIMARY_FCB
379 1368 3 THEN KERNEL_CALL (INIT_FCB2, .FCB, .HEADER);
380 1369 3
381 1370 3 ! Compute the number of blocks being deallocated and credit them to the
382 1371 3 ! file owner. We compute this by taking the total space mapped by the header,
383 1372 3 ! less the number of blocks passed over in the scan.
384 1373 3
385 1374 3
386 1375 3 IF NOT .CLEANUP_FLAGS[CLF_NOTCHARGED]
387 1376 3 THEN
388 1377 3 BEGIN
389 1378 3 HEADER_SIZE = FILE_SIZE (.ALT_HEADER);
390 1379 3 CHARGE_QUOTA (.HEADER[FH2$L_FILEOWNER], - (.HEADER_SIZE - .HEADER_VBN),
391 1380 3 BITLIST (QUOTA_CHARGE));
392 1381 3 END;
393 1382 3
394 1383 3 ! Now we can free the blocks being truncated off. They are turned over
395 1384 3 ! either to the storage map, or to the bad block file.
396 1385 3
397 1386 3
398 1387 3 IF .FIB[FIB$V_MARKBAD]
399 1388 3 THEN
400 1389 3 DEALLOCATE_BAD (.FIB, .ALT_HEADER, .TRUNC_POINTER, .VBN)
401 1390 3 ELSE
402 1391 3 TRUNCATE_HEADER (.FIB, .ALT_HEADER, .TRUNC_POINTER, .VBN);
```



```

403      1392 3
404      1393 3 REREAD2 = .REREAD;
405      1394 3 IF .EXT_FID[FID$W_NUM] NEQ 0
406      1395 3 OR .EXT_FID[FID$W_RVN] NEQ 0
407      1396 3 THEN
408      1397 4 BEGIN
409      1398 4 REREAD = 1;
410      1399 4 HEADER = NEXT HEADER (0, .FCB, EXT_FID, .EX_SEGNUM);
411      1400 4 KERNEL_CALL (DEL_EXTFCB, .FCB);
412      1401 4 DELETE_FILE (.FIB, .HEADER);
413      1402 3 END;
414      1403 3
415      1404 2 END; ! end of block DO_TRUNCATE
416      1405 2
417      1406 2 ! If this was a truncate of a multi-header file, reread the primary header
418      1407 2 ! and update the HIBLK field in the record attributes to reflect the new file
419      1408 2 ! size.
420      1409 2
421      1410 2
422      1411 2 IF .REREAD
423      1412 2 THEN HEADER = READ_HEADER (0, .PRIMARY_FCB);
424      1413 2
425      1414 2 IF .REREAD2
426      1415 2 THEN
427      1416 3 BEGIN
428      1417 3 BBLOCK [HEADER[FH2$W_RECATTR], FAT$L_HIBLK] = ROT (.FIB[FIB$L_EXVBN]-1, 16);
429      1418 3 MARK_DIRTY (.HEADER);
430      1419 3 END;
431      1420 2
432      1421 2
433      1422 2 ! Stop metering of this subfunction
434      1423 2
435      1424 2
436      1425 2 PMS_END_SUB ();
437      1426 2
438      1427 1 END; ! end of routine TRUNCATE
```

```
.TITLE TRUNC
.IDENT \V04-000\
```

```
.EXTRN PMS_START_SUB, PMS_END_SUB
.EXTRN FILE_SIZE, SEARCH_FCB
.EXTRN CHARGE_QUOTA, DEALLOCATE_BAD
.EXTRN MARK_DIRTY, CHECKSUM
.EXTRN GET_MAP_POINTER
.EXTRN MAKE_POINTER, NEXT_HEADER
.EXTRN CREATE_BLOCK, INVALIDATE
.EXTRN INIT_FCB2, WRITE_HEADER
.EXTRN READ_HEADER, DEL_EXTFCB
.EXTRN DELETE_FILE
```

```
.PSECT $CODE$,NOWRT,2
```

```
.ENTRY TRUNCATE, Save R2,R3,R4,R5,R6,R7,R8,R9,R11 ; 1107
SUBL2 #36, SP ;
MOVAB -128(BASE), R2 ; 1178
```

```
OBFC 00000
5E 24 C2 00002
52 80 AA 9E 00005
```

		0000G	CF	08	DD	00009	PUSHL	#8	1207
			50	01	FB	0000B	CALLS	#1, PMS_START_SUB	
				04	AC	7D	MOVQ	FIB, R0	1209
					0000V	30	BSBW	TRUNC CHECKS	
		10	AE	08	AC	D0	MOVL	FILEHEADER, HEADER	1216
08	AE	OC	AE	08	AA	D0	MOVL	8(BASE), FCB	1217
		OC	AC		01	C3	SUBL3	#1, TRNVBN, VBN	1218
			50	04	AC	D0	MOVL	FIB, R0	1223
		1C	A0		01	D0	MOVL	#1, 28(R0)	
				04	AE	D4	CLRL	REREAD	1225
	51	10	AE		01	C1	ADDL3	#1, HEADER, R1	1239
			50		61	9A	MOVZBL	(R1), R0	
			58	10	BE40	3E	MOVAW	@HEADER[R0], MAP_POINTER	
	51	10	AE		3A	C1	ADDL3	#58, HEADER, R1	1240
			50		61	9A	MOVZBL	(R1), R0	
			5B		6840	3E	MOVAW	(MAP_POINTER)[R0], MAP_END	
			59		58	D0	MOVL	MAP_POINTER, PREV_POINTER	1241
			6E	08	AE	D0	MOVL	VBN, HEADER_VBN	1242
			5B		58	D1	CMPL	MAP_POINTER, MAP_END	1244
					1E	1E	BGEQU	3\$	
					0000G	30	BSBW	GET MAP_POINTER	1246
		08	AE		56	D1	CMPL	COUNT, VBN	1247
					47	1E	BGEQU	5\$	
		08	AE		56	C2	SUBL2	COUNT, VBN	1248
			50	04	AC	D0	MOVL	FIB, R0	1249
		1C	A0		56	C0	ADDL2	COUNT, 28(R0)	
					56	D5	TSTL	COUNT	1250
					E2	13	BEQL	2\$	
			59		58	D0	MOVL	MAP_POINTER, PREV_POINTER	1251
					DD	11	BRB	2\$	1244
				OC	AE	DD	PUSHL	FCB	1259
				14	AE	DD	PUSHL	HEADER	
		0000G	CF		02	FB	CALLS	#2, NEXT_HEADER	
			53		50	D0	MOVL	R0, NEW_HEADER	
					18	13	BEQL	4\$	1260
		04	AE		01	D0	MOVL	#1, REREAD	1261
		10	AE		53	D0	MOVL	NEW_HEADER, HEADER	1262
				OC	AE	D5	TSTL	FCB	1264
					A0	13	BEQL	1\$	
	50	OC	AE		OC	C1	ADDL3	#12, FCB, R0	1265
		OC	AE		60	D0	MOVL	(R0), FCB	
					95	11	BRB	1\$	1237
				08	AE	D5	TSTL	VBN	1268
					05	13	BEQL	5\$	
				0870	8F	BF	CHMU	#2160	1269
					04	00A6	RET		
		04	A2	08	AE	CE	MNEGL	VBN, 4(R2)	1278
			50	04	AC	D0	MOVL	FIB, R0	1280
	OC	17	A0		02	E0	BBS	#2, 23(R0), 6\$	
			50		98	AA	MOVL	-104(BASE), R0	1281
			51		3C	A0	MOVZWL	60(R0), R1	
					51	D7	DECL	R1	
					02	11	BRB	7\$	
					51	D4	CLRL	R1	1280
			51	08	AE	C0	ADDL2	VBN, R1	
			50	98	AA	D0	MOVL	-104(BASE), R0	1284
			53	3C	A0	3C	MOVZWL	60(R0), R3	



08	AE	04	51	3C	53	C6	000CF	DIVL2	R3, R1	:		
			54		A0	3C	000D2	MOVZWL	60(R0), R4	:		
			51		54	C5	000D6	MULL3	R4, R1, VBN	:		
		04	A2	08	AE	C0	000DB	ADDL2	VBN, 4(R2)	:	1285	
			50	04	AC	D0	000E0	MOVL	FIB, R0	:	1286	
		1C	A0	08	AE	C0	000E4	ADDL2	VBN, 28(R0)	:		
			6E	04	A2	C0	000E9	ADDL2	4(R2), HEADER_VBN	:	1287	
			56	08	AE	D1	000ED	CMPL	VBN, COUNT	:	1295	
					20	12	000F1	BNEQ	8\$	:		
				08	AE	D4	000F3	CLRL	VBN	:	1298	
			59		58	D0	000F6	MOVL	MAP_POINTER, PREV_POINTER	:	1299	
			5B		59	D1	000F9	CMPL	PREV_POINTER, MAP_END	:	1300	
					15	1F	000FC	BLSSU	8\$	:		
	50	10	AE		0E	C1	000FE	ADDL3	#14, HEADER, R0	:	1301	
					60	B5	00103	TSTW	(R0)	:		
	50	10	AE		0C	12	00105	BNEQ	8\$	:		
					12	C1	00107	ADDL3	#18, HEADER, R0	:	1302	
					60	B5	0010C	TSTW	(R0)	:		
					03	12	0010E	BNEQ	8\$	:		
			50	04	019E	31	00110	BRW	21\$	:		
	2B	17	A0		AC	D0	00113	MOVL	FIB, R0	:	1313	
			5B		02	E1	00117	BBC	#2, 23(R0), 10\$	:		
					58	D1	0011C	CMPL	MAP_POINTER, MAP_END	:	1315	
					23	12	0011F	BNEQ	9\$	:		
	51		56	08	AE	C3	00121	SUBL3	VBN, COUNT, R1	:	1316	
51	3C	A0	50	98	AA	D0	00126	MOVL	-104(BASE), R0	:		
			10		00	ED	0012A	CMPZV	#0, #16, 60(R0), R1	:		
					12	12	00130	BNEQ	9\$	:		
	50	10	AE		0E	C1	00132	ADDL3	#14, HEADER, R0	:	1317	
					60	B5	00137	TSTW	(R0)	:		
					09	12	00139	BNEQ	9\$	:		
	50	10	AE		10	C1	0013B	ADDL3	#16, HEADER, R0	:	1318	
					60	B5	00140	TSTW	(R0)	:		
					03	13	00142	BEQL	10\$	:		
					14	BF	00144	CHMU	#20	:	1319	
						04	00146	RET		:		
			6A		12	88	00147	BISB2	#18, (BASE)	:	1325	
			50	08	AA	D0	0014A	MOVL	8(BASE), R0	:	1326	
			51	04	AC	D0	0014E	MOVL	FIB, R1	:		
	38	A0	1C		01	C3	00152	SUBL3	#1, 28(R1), 56(R0)	:		
			12	04	AE	E8	00158	BLBS	REREAD, 11\$	:	1332	
			50	04	AC	D0	0015C	MOVL	FIB, R0	:	1333	
					01	C3	00160	SUBL3	#1, 28(R0), R0	:		
	50	1C	A0		18	C1	00165	ADDL3	#24, HEADER, R1	:		
	51	10	AE		10	9C	0016A	ROTL	#16, R0, (R1)	:		
	61		50		01	7D	0016E	MOVQ	#1, -(SP)	:	1341	
			7E		01	CE	00171	MNEGL	#1, -(SP)	:		
			7E		03	FB	00174	CALLS	#3, CREATE BLOCK	:		
		0000G	CF		50	D0	00179	MOVL	R0, ALT HEADER	:		
		14	AE	14	AE	DD	0017D	PUSHL	ALT HEADER	:	1342	
					01	FB	00180	CALLS	#1, INVALIDATE	:		
	14	BE	0000G	0200	8F	28	00185	MOVC3	#512, @HEADER, @ALT HEADER	:	1343	
		50		10	AE	C3	0018D	SUBL3	HEADER, PREV_POINTER, R0	:	1344	
					14	BE40	9E	00192	MOVAB	@ALT HEADER[R0], TRUNC_POINTER	:	
					02	C6	00198	DIVL2	#2, R0	:	1346	
	51	10	AE		3A	C1	0019B	ADDL3	#58, HEADER, R1	:		
	52	10	AE		01	C1	001A0	ADDL3	#1, HEADER, R2	:		



61		50	08	62	83	001A5	SUBB3	(R2), R0, (R1)		
				AE	D5	001A9	TSTL	VBN		1347
		58		21	13	001AC	BEQL	14\$		
				59	D0	001AE	MOVL	PREV_POINTER, MAP_POINTER		1350
				0000G	30	001B1	BSBW	GET_MAP_POINTER		1351
	C0	8F	01	A9	93	001B4	BITB	1(PREV_POINTER), #192		1353
		7E		05	12	001B9	BNEQ	12\$		
				69	3C	001BB	MOVZWL	(PREV_POINTER), -(SP)		1354
				02	11	001BE	BRB	13\$		
				7E	D4	001C0	CLRL	-(SP)		1353
			14	AE	DD	001C2	PUSHL	HEADER		1352
				57	DD	001C5	PUSHL	LBN		
			14	AE	DD	001C7	PUSHL	VBN		
				04	FB	001CA	CALLS	#4, MAKE_POINTER		
	51	0000G		02	C1	001CF	ADDL3	#2, HEADER, R1		1357
	10	AE		61	9A	001D4	MOVZBL	(R1), R0		
		50	10	BE40	3E	001D7	MOVAV	@HEADER[R0], MAP_END		
		5B		5B	D1	001DC	CMPL	MAP_END, PREV_POINTER		1358
		59		09	15	001DF	BLEQ	15\$		
				59	C2	001E1	SUBL2	PREV_POINTER, R11		1359
5B	00	5B		00	2C	001E4	MOVC5	#0, (SP), #0, R11, (PREV_POINTER)		
		6E		69		001E9				
	50	10		04	C1	001EA	ADDL3	#4, HEADER, R0		1361
		AE		60	3C	001EF	MOVZWL	(R0), EX_SEGNUM		
		56		56	D6	001F2	INCL	EX_SEGNUM		
	57	10		0E	C1	001F4	ADDL3	#14, HEADER, R7		1362
	AE	AE		06	28	001F9	MOVC3	#6, (R7), EXT_FID		
	57	10		0E	C1	001FE	ADDL3	#14, HEADER, R7		1363
06	00	AE		00	2C	00203	MOVC5	#0, (SP), #0, #6, (R7)		
		6E		67		00208				
			10	AE	DD	00209	PUSHL	HEADER		1364
				01	FB	0020C	CALLS	#1, CHECKSUM		
	0000G	CF		00	FB	00211	CALLS	#0, WRITE_HEADER		1365
	0000G	CF		0C	AE	D5	TSTL	FCB		1367
				12	13	00219	BEQL	16\$		
	08	AA		0C	AE	D1	CMPL	FCB, 8(BASE)		
				0B	13	00220	BEQL	16\$		
			10	AE	DD	00222	PUSHL	HEADER		1368
			10	AE	DD	00225	PUSHL	FCB		
	0000G	CF		02	FB	00228	CALLS	#2, INIT_FCB2		
1B		6A		1D	E0	0022D	BBS	#29, (BASE), 17\$		1375
			14	AE	DD	00231	PUSHL	ALT_HEADER		1378
	0000G	CF		01	FB	00234	CALLS	#1, FILE_SIZE		
				02	DD	00239	PUSHL	#2		1380
	7E	04		50	C3	0023B	SUBL3	HEADER_SIZE, HEADER_VBN, -(SP)		1379
	52	18		3C	C1	00240	ADDL3	#60, HEADER, R2		
				62	DD	00245	PUSHL	(R2)		
	0000G	CF		03	FB	00247	CALLS	#3, CHARGE_QUOTA		
		50	04	AC	D0	0024C	MOVL	FIB, R0		1387
	14	17		02	E1	00250	BBC	#2, 23(R0), 18\$		
		AO	08	AE	DD	00255	PUSHL	VBN		1389
			1C	AE	DD	00258	PUSHL	TRUNC_POINTER		
			1C	AE	DD	0025B	PUSHL	ALT_HEADER		
				50	DD	0025E	PUSHL	R0		
	00000000G	00		04	FB	00260	CALLS	#4, DEALLOCATE_BAD		
				10	11	00267	BRB	19\$		
			08	AE	DD	00269	PUSHL	VBN		1391



TRUNC  
V04-000

H 11  
16-Sep-1984 01:19:12  
14-Sep-1984 12:30:50

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[F11X.SRC]TRUNC.B32;1

Page 13  
(2)

		1C	AE	DD	0026C	PUSHL	TRUNC_POINTER	:	
		1C	AE	DD	0026F	PUSHL	ALT_HEADER	:	
			50	DD	00272	PUSHL	R0	:	
0000V	CF		04	FB	00274	CALLS	#4, TRUNCATE_HEADER	:	
	52	04	AE	D0	00279	19%: MOVL	REREAD, REREAD2	:	1393
		1C	AE	B5	0027D	TSTW	EXT_FID	:	1394
			05	12	00280	BNEQ	20\$	:	
		20	AE	B5	00282	TSTW	EXT_FID+4	:	1395
			2A	13	00285	BEQL	21\$	:	
04	AE		01	D0	00287	20%: MOVL	#1, REREAD	:	1398
			56	DD	0028B	PUSHL	EX_SEGNUM	:	1399
		20	AE	9F	0028D	PUSHAB	EXT_FID	:	
		14	AE	DD	00290	PUSHL	FCB	:	
			7E	D4	00293	CLRL	-(SP)	:	
0000G	CF		04	FB	00295	CALLS	#4, NEXT_HEADER	:	
10	AE		50	D0	0029A	MOVL	R0, HEADER	:	
		0C	AE	DD	0029E	PUSHL	FCB	:	1400
0000G	CF		01	FB	002A1	CALLS	#1, DEL_EXTFCB	:	
		10	AE	DD	002A6	PUSHL	HEADER	:	1401
		04	AC	DD	002A9	PUSHL	FIB	:	
0000G	CF		02	FB	002AC	CALLS	#2, DELETE_FILE	:	
	OE	04	AE	E9	002B1	21%: BLBC	REREAD, 22\$	:	1411
		08	AA	DD	002B5	PUSHL	8(BASE)	:	1412
			7E	D4	002B8	CLRL	-(SP)	:	
0000G	CF		02	FB	002BA	CALLS	#2, READ_HEADER	:	
10	AE		50	D0	002BF	MOVL	R0, HEADER	:	
	1A		52	E9	002C3	22%: BLBC	REREAD2, 23\$	:	1414
	50	04	AC	D0	002C6	MOVL	FIB, R0	:	1417
50	1C		01	C3	002CA	SUBL3	#1, 28(R0), R0	:	
51	10		18	C1	002CF	ADDL3	#24, HEADER, R1	:	
61			10	9C	002D4	ROTL	#16, R0, (R1)	:	
		10	AE	DD	002D8	PUSHL	HEADER	:	1418
0000G	CF		01	FB	002DB	CALLS	#1, MARK_DIRTY	:	
0000G	CF		00	FB	002E0	23%: CALLS	#0, PMS_END_SUB	:	1425
			04	002E5	RET			:	1427

; Routine Size: 742 bytes, Routine Base: \$CODE\$ + 0000

W1  
V0

```

440 1428 1 GLOBAL ROUTINE TRUNCATE_HEADER (FIB, HEADER, POINTER, LAST_COUNT) : L_NORM NOVALUE =
441 1429 1
442 1430 1 ++
443 1431 1
444 1432 1 FUNCTIONAL DESCRIPTION:
445 1433 1
446 1434 1 This routine returns the indicated retrieval pointers in the given
447 1435 1 file header to the storage map.
448 1436 1
449 1437 1
450 1438 1 CALLING SEQUENCE:
451 1439 1 TRUNCATE_HEADER (ARG1, ARG2, ARG3, ARG4)
452 1440 1
453 1441 1 INPUT PARAMETERS:
454 1442 1 ARG1: address of FIB of operation
455 1443 1 ARG2: address of file header
456 1444 1 ARG3: address of first retrieval pointer to process, if present
457 1445 1 ARG4: new count field of first pointer, if present
458 1446 1
459 1447 1 IMPLICIT INPUTS:
460 1448 1 NONE
461 1449 1
462 1450 1 OUTPUT PARAMETERS:
463 1451 1 NONE
464 1452 1
465 1453 1 IMPLICIT OUTPUTS:
466 1454 1 NONE
467 1455 1
468 1456 1 ROUTINE VALUE:
469 1457 1 NONE
470 1458 1
471 1459 1 SIDE EFFECTS:
472 1460 1 file header altered, storage map altered
473 1461 1
474 1462 1 --
475 1463 1
476 1464 2 BEGIN
477 1465 2
478 1466 2 MAP
479 1467 2 FIB : REF BBLOCK, ! user FIB
480 1468 2 HEADER : REF BBLOCK; ! file header
481 1469 2
482 1470 2 GLOBAL REGISTER
483 1471 2 COUNT = 6, ! count of blocks returned
484 1472 2 LBN = 7, ! LBN of map entry
485 1473 2 MAP_POINTER = 8 : REF BBLOCK; ! pointer to scan map
486 1474 2
487 1475 2 LOCAL
488 1476 2 MAP_END; ! address of end of map area
489 1477 2
490 1478 2 BIND_COMMON;
491 1479 2
492 1480 2 EXTERNAL ROUTINE
493 1481 2 GET_MAP_POINTER : L_MAP_POINTER, ! get value of next map entry
494 1482 2 RETURN_BLOCKS : L_NORM ADDRESSING_MODE (GENERAL);
495 1483 2 ! return blocks to storage map
496 1484 2
```



```

497 1485 2
498 1486 2 LOCAL
499 1487 2 ERASE_FLAG;
500 1488 2
501 1489 2 ! Determine if blocks being returned should be erased. Erase them if
502 1490 2 ! either the volume or file erase attribute is set.
503 1491 2
504 1492 2 ERASE_FLAG = 0; ! Assume no erase necessary
505 1493 2 IF .CURRENT_VCB[VCBSV_ERASE] ! Check the volume attribute
506 1494 2 OR .HEADER[FH2$V_ERASE] ! Check the file attribute in the header
507 1495 2 THEN
508 1496 2 ERASE_FLAG = 1
509 1497 2 ELSE
510 1498 2 IF .PRIMARY_FCB NEQ 0 ! Check the file attribute in the FCB
511 1499 2 THEN
512 1500 2 IF .PRIMARY_FCB[FCBSV_ERASE]
513 1501 2 THEN
514 1502 2 ERASE_FLAG = 1;
515 1503 2
516 1504 2 !
517 1505 2 ! Establish pointers into the file header. If explicit args are supplied, use
518 1506 2 ! them; else default to releasing the entire file header.
519 1507 2 !
520 1508 2
521 1509 2 MAP_POINTER = .HEADER + .HEADER[FH2$B_MPOFFSET]*2;
522 1510 2 MAP_END = .MAP_POINTER + .HEADER[FH2$B_MAP_INUSE]*2;
523 1511 2
524 1512 2 IF ACTUALCOUNT GEQ 4
525 1513 2 THEN
526 1514 2 BEGIN
527 1515 2 MAP_POINTER = .POINTER;
528 1516 2 IF .LAST_COUNT NEQ 0
529 1517 2 THEN
530 1518 2 BEGIN
531 1519 2 GET MAP_POINTER ();
532 1520 2 RETURN_BLOCKS (.LBN+.LAST_COUNT, .COUNT-.LAST_COUNT, .ERASE_FLAG);
533 1521 2 FIB[FIB$L_EXSZ] = .FIB[FIB$L_EXSZ] + .COUNT - .LAST_COUNT;
534 1522 2 END;
535 1523 2 END;
536 1524 2
537 1525 2 ! Now scan the map area, cleaning out pointers and releasing blocks.
538 1526 2 !
539 1527 2
540 1528 2 UNTIL .MAP_POINTER GEQA .MAP_END DO
541 1529 2 BEGIN
542 1530 2 GET MAP_POINTER ();
543 1531 2 RETURN_BLOCKS (.LBN, .COUNT, .ERASE_FLAG);
544 1532 2 FIB[FIB$L_EXSZ] = .FIB[FIB$L_EXSZ] + .COUNT;
545 1533 2 END;
546 1534 2
547 1535 1 END; ! end of routine TRUNCATE_HEADER
```

.EXTRN RETURN\_BLOCKS

01DC 00000

.ENTRY TRUNCATE\_HEADER, Save R2,R3,R4,R6,R7,R8

; 1428

		54	00000000G	00	9E	00002	MOVAB	RETURN_BLOCKS, R4	:	1492
				52	D4	00009	CLRL	ERASE_FLAG	:	1493
		50		98	AA	D0 0000B	MOVL	-104(BASE), R0	:	1494
14		A0		03	E0	0000F	BBS	#3, 83(R0), 1\$	:	1498
		50		08	AC	D0 00014	MOVL	HEADER, R0	:	1499
0B		A0		01	E0	00018	BBS	#1, 54(R0), 1\$	:	1500
		50		08	AA	D0 0001D	MOVL	8(BASE), R0	:	1502
				08	13	00021	BEQL	2\$	:	1509
03		A0		06	E1	00023	BBC	#6, 34(R0), 2\$	:	1510
		52		01	D0	00028	MOVL	#1, ERASE_FLAG	:	1502
		51		08	AC	D0 0002B	MOVL	HEADER, R1	:	1509
		50		01	A1	9A 0002F	MOVZBL	1(R1), R0	:	1510
		58			6140	3E 00033	MOVAV	(R1)[R0], MAP_POINTER	:	1511
		50		3A	A1	9A 00037	MOVZBL	58(R1), R0	:	1512
		53			6840	3E 0003B	MOVAV	(MAP_POINTER)[R0], MAP_END	:	1515
		04			6C	91 0003F	CMPB	(AP), #4	:	1516
				29	1F	00042	BLSSU	3\$	:	1519
		58		0C	AC	D0 00044	MOVL	POINTER, MAP_POINTER	:	1520
				10	AC	D5 00048	TSTL	LAST_COUNT	:	1521
				20	13	0004B	BEQL	3\$	:	1528
					0000G	30 0004D	BSBW	GET_MAP_POINTER	:	1530
				52	DD	00050	PUSHL	ERASE_FLAG	:	1531
		56		10	AC	C3 00052	SUBL3	LAST_COUNT, COUNT, -(SP)	:	1532
7E				10	BC47	9F 00057	PUSHAB	@LAST_COUNT[LBN]	:	1532
		64			03	FB 0005B	CALLS	#3, RETURN_BLOCKS	:	1528
		50		04	AC	D0 0005E	MOVL	FIB, R0	:	1528
		56		18	A0	C1 00062	ADDL3	24(R0), COUNT, R1	:	1528
18		51		10	AC	C3 00067	SUBL3	LAST_COUNT, R1, 24(R0)	:	1528
		53			58	D1 0006D	CMPB	MAP_POINTER, MAP_END	:	1528
				16	1E	00070	BGEQU	4\$	:	1530
					0000G	30 00072	BSBW	GET_MAP_POINTER	:	1531
				52	DD	00075	PUSHL	ERASE_FLAG	:	1532
				56	DD	00077	PUSHL	COUNT	:	1532
				57	DD	00079	PUSHL	LBN	:	1532
		64			03	FB 0007B	CALLS	#3, RETURN_BLOCKS	:	1532
		50		04	AC	D0 0007E	MOVL	FIB, R0	:	1528
18		A0			56	C0 00082	ADDL2	COUNT, 24(R0)	:	1528
					E5	11 00086	BRB	3\$	:	1535
					04	00088	RET		:	1535

; Routine Size: 137 bytes, Routine Base: \$CODE\$ + 02E6

; 548 1536 1



```

: 550      1537 1 GLOBAL ROUTINE TRUNC_CHECKS (FIB, HEADER) : L_JSB_2ARGS NOVALUE =
: 551      1538 1
: 552      1539 1
: 553      1540 1
: 554      1541 1
: 555      1542 1
: 556      1543 2 BEGIN
: 557      1544 2
: 558      1545 2 MAP
: 559      1546 2     FIB      : REF BBLOCK,
: 560      1547 2     HEADER : REF BBLOCK;
: 561      1548 2
: 562      1549 2 BIND_COMMON;
: 563      1550 2
: 564      1551 2 ! The block count must be zero (default).
: 565      1552 2 ! If the operation calls for the blocks to be turned over to the bad block
: 566      1553 2 ! file, the caller must be system.
: 567      1554 2
: 568      1555 2
: 569      1556 2 IF .FIB[FIB$V MARKBAD]
: 570      1557 2 AND NOT .CLEANUP_FLAGS[CLF_SYSPRV]
: 571      1558 2 THEN ERR_EXIT (SS$_NOPRIV);
: 572      1559 2
: 573      1560 2 ! Check for the index file INDEXF.SYS
: 574      1561 2
: 575      1562 2
: 576      1563 2 IF .HEADER[FH2$W_FID_NUM] EQL FID$_INDEXF
: 577      1564 2 AND .HEADER[FH2$W_FID_SEQ] EQL FID$_INDEXF
: 578      1565 2 AND .HEADER[FH2$B_FID_NMX] EQL 0
: 579      1566 2 THEN ERR_EXIT (SS$_NOPRIV);
: 580      1567 2
: 581      1568 2 IF .FIB[FIB$L_EXSZ] NEQ 0 THEN ERR_EXIT (SS$_BADPARAM);
: 582      1569 2
: 583      1570 2 ! Init the user's return parameters.
: 584      1571 2
: 585      1572 2
: 586      1573 2 FIB[FIB$L_EXSZ] = 0;
: 587      1574 2
: 588      1575 1 END;
```

04	17	A0	02	E1	00000	TRUNC_CHECKS::			
		11	01	AA	E9	00005	BBC	#2, 23(FIB), 1\$	: 1556
		01	08	A1	B1	00009 1\$:	BLBC	1(BASE), 2\$	: 1557
				0E	12	0000D	CMPW	8(HEADER), #1	: 1563
		01	0A	A1	B1	0000F	BNEQ	3\$	
				08	12	00013	CMPW	10(HEADER), #1	: 1564
				0D	A1	95 00015	BNEQ	3\$	
					03	12 00018	TSTB	13(HEADER)	: 1565
					24	BF 0001A 2\$:	BNEQ	3\$	
						05 0001C	CHMU	#36	: 1566
			18	A0	D5	0001D 3\$:	RSB		
				03	13	00020	TSTL	24(FIB)	: 1568
							BEQL	4\$	:

TRUNC  
V04-000

M 11  
16-Sep-1984 01:19:12  
14-Sep-1984 12:30:50

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[F11X.SRC]TRUNC.B32;1 Page 18  
(4)

14 BF 00022 CHMU #20  
05 00024 RSB  
18 A0 D4 00025 4\$: CLRL 24(FIB)  
05 00028 RSB

:  
:  
: 1573  
: 1575

; Routine Size: 41 bytes, Routine Base: \$CODE\$ + 036F

: 589 1576 1  
: 590 1577 1 END  
: 591 1578 0 ELUDOM

#### PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	920	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

#### Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	49	0	1000	00:01.9

#### COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:TRUNC/OBJ=OBJ\$:TRUNC MSRC\$:TRUNC/UPDATE=(ENH\$:TRUNC)

; Size: 920 code + 0 data bytes  
; Run Time: 00:42.4  
; Elapsed Time: 01:32.4  
; Lines/CPU Min: 2230  
; Lexemes/CPU-Min: 52730  
; Memory Used: 353 pages  
; Compilation Complete



0173 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

SCHFCB  
LIS

SND5MB  
LIS

SHFDLR  
LIS

SNDERL  
LIS

TRUNC  
LIS

FAL

FAL  
MAP

SELVOL  
LIS

DAPDEF  
MOL

SMALOC  
LIS

SNOBAD  
LIS

SWITUL  
LIS

WITURN  
LIS